

SART: Speeding up Query Processing in Sensor Networks with an Autonomous Range Tree Structure

Spyros Sioutas
Department of Informatics,
Ionian University
49100 Corfu, Greece
sioutas@ionio.gr

Dimitrios Tsoumakos
Department of Informatics,
Ionian University
49100 Corfu, Greece
dtsouma@ionio.gr

Alexandros Panaretos
Department of Informatics,
Ionian University
49100 Corfu, Greece
alex@ionio.gr

Giannis Tzimas
Dept. of Applied Informatics in
Management and Economy,
Techn. Educ. Institute
30200 Messolonghi, Greece
tzimas@teimes.gr

Ioannis Karydis
Department of Informatics,
Ionian University
49100 Corfu, Greece
karydis@ionio.gr

Dimitrios Tsolis
Cult. Herit. Management and
New Technologies Dept.,
University of Western Greece
30100 Agrinio, Greece
dtsolis@upatras.gr

ABSTRACT

We consider the problem of constructing efficient P2P overlays for sensor networks providing "Energy-Level Application and Services". In this context, assuming that a sensor is responsible for executing some program task but unfortunately its energy-level is lower than a pre-defined threshold. Then, this sensor should be able to introduce a query to the whole system in order to discover efficiently another sensor with the desired energy level, in which the task overhead must be eventually forwarded. In this way, the "Life-Expectancy" of the whole network could be increased. Sensor nodes are mapped to peers based on their energy level. As the energy levels change, the sensor nodes would have to move from one peer to another and this operation is very crucial for the efficient scalability of the proposed system. Similarly, as the energy level of a sensor node becomes extremely low, that node may want to forward its task to another node with the desired energy level. The method presented in [15] presents a novel P2P overlay for Energy Level discovery in a sensor network. However, this solution is not dynamic, since requires periodical restructuring. In particular, it is not able to support neither join of sensor nodes with energy level out of the ranges supported by the existing p2p overlay nor leave of *empty* overlay-peers to which no sensor nodes are currently associated. On this purpose and based on the efficient P2P method presented in [16], we design a dynamic P2P overlay for Energy Level discovery in a sensor network, the so-called SART (Sensors' Autonomous Range Tree) ¹. The

adaptation of the P2P index presented in [16] guarantees the best-known dynamic query performance of the above operation. We experimentally verify this performance, via the D-P2P-Sim simulator ².

1. INTRODUCTION

In the last years sensor network research primarily focused on data collection, finding applications in ecology (e.g., environmental and habitat monitoring [13]), in precision agriculture (e.g., monitoring of temperature and humidity), in civil engineering (e.g., monitoring stress levels of buildings under earthquake simulations), in military and surveillance (e.g., tracking of an intruder [7]), in aerospace industry (e.g., fairing of cargo in a rocket), etc.

Traditionally, sensors are used as data gathering instruments, which continuously feed a central base station database. The queries are executed in this centralized base station database which continuously collates the data. However, given the current trends (increase in numbers of sensors, together collecting gigabits of data, increase in processing power at sensors) it is not anymore feasible to use a centralized solution for querying the sensor networks. Therefore, there is a need for establishing an efficient access structure on sensor networks in order to contact only the relevant nodes for the execution of a query and hence achieve minimal energy consumption, minimal response time, and an accurate response. We achieve these goals with our peer-to-peer query processing model on top of a distributed index structure on wireless sensor networks.

In sensor networks any node should be able to introduce a query to the system. For example, in the context of a fire evacuation scenario a firefighter should be able to query a nearby sensor node for the closest exit where safe paths exist. Therefore, a peer-to-peer query processing model is required. A first P2P program for spatial query execution presented in [8].

¹Preliminary version of this work was presented in ACM SAC'12, pp.847-852, March 25-29, 2012, Riva del Garda, Italy

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

²D-P2P-Sim is publicly available at <http://code.google.com/p/d-p2p-sim/>

According to [1], the benefits of the P2P overlays in sensor networks are the following: Efficient Data Lookup, Guaranties on Lookup Times, Location Independence, Overlay Applications and Services, Elimination of proxies/sinks with undesirable central authority, Limited Broadcast. P2P design, for Internet-like environments, has been a very active research area and there are many P2P Internet protocols and systems available like CAN [3], Pastry [3], and Chord [3]. The main arguments against P2P designs in sensor networks were the following: Logical Topology=Physical Topology, Route Maintenance Overhead, Sensor Nodes are Not Named, DHTs are Computationally Intensive. By overcoming the arguments above (for details see [1], [2] and [4]), in [2] and [4] the first DHT (Distributed Hash Table) based protocols for sensor networks were presented, the CSN and VRR respectively. In [1] the Tiered Chord (TChord) protocol was proposed, which is similar to, and inspired by, CSN. TChord is a simplified mapping of Chord onto sensor networks. Unlike CSN the design of TChord is more generic (to support a variety of applications and services on top instead of just serving incoming data queries). Gerla et al. argue for the applicability and transfer of wired P2P models and techniques to MANETs [9].

Most existing decentralized discovery solutions in practice are either DHT based, like Chord or hierarchical clustering based, like BATON [3], NBDT [14], ART [16] or Skip-Graphs [3]. The majority of existing P2P overlays for sensor networks were designed in a DHT fashion and the best current solution is the TChord. On the contrary, ELDT [15] is the only existing P2P protocol for sensor networks, which combines the benefits of both DHT and hierarchical [14] clustering fashions. In this solution, sensor nodes are mapped to peers based on their energy level. As the energy levels change, the sensor nodes would have to move from one peer to another and this operation is very crucial for the efficient scalability of the proposed system. Similarly, as the energy level of a sensor node becomes extremely low, that node may want to forward its task to another node with the desired energy level. However, the ELDT solution is not dynamic, since requires periodical restructuring. In particular, it is not able to support neither join of sensor nodes with energy level out of the ranges supported by the existing p2p overlay nor leave of *empty* overlay-peers to which no sensor nodes are currently associated. On this purpose and based on the efficient P2P method presented in [16], we design a dynamic P2P overlay for Energy Level discovery in a sensor network, the so-called SART (Sensors' Autonomous Range Tree). The adaptation of the P2P index presented in [16] guarantees the best-known dynamic query performance of the above operation.

The main functionalities of SART attempt to increase the "Life-Expectancy" of the whole sensor network in dynamic way, providing support for processing: (a) exact match queries of the form "given a sensor node with low energy-level k' , locate a sensor node with high energy-level k , where $k \gg k'$ " (the task will be forwarded to the detected sensor node) (b) range queries of the form "given an energy-level range $[k, k']$, locate the sensor node/nodes the energy-levels of which belong to this range" (the task will be forwarded to one of the detected sensor nodes) (c) update queries of the form "find the new overlay-peer to which the sensor node must be moved (or associated) according to its current energy level" (the energy level of each sensor node is a decreas-

ing function of time and utilization) (d) join queries of the form "join a new overlay-peer to which the new (inserted) sensor node is associated" and (e) leave queries of the form "leave (delete) the overlay-peer to which no sensor nodes are currently associated". The SART overlay adapts the novel idea of ART P2P infrastructure presented in [16] providing functionalities in optimal time. For comparison purposes, an elementary operation's evaluation is presented in table 1 between ART, NBDT, Skip-Graphs [3], Chord [3] and its newest variation (F-Chord [3]), BATON and its newest variation (BATON* [3]). The rest of this paper is structured as follows. Section 2 and 3 describe the SART system while section 4 presents an extended experimental verification via an appropriate simulator we have designed for this purpose. Section 5 concludes.

2. THE SART PROTOCOL

SART, is a simplified mapping of ART [16] onto sensor networks. Like ART, at the heart of SART, lookup and join/leave respectively are the two main operations. Given a set of sensor nodes, we hash the unique address of each sensor node to obtain node identifiers. Meta-data keys, generated from the data stored on the nodes, are hashed to obtain key identifiers.

The SART protocol (see figure 1) is an hierarchical arrangement of some sensor nodes (master nodes). The master node of level i maintains information (in its local finger table) about all its *slave nodes* and 2^{i-1} other master nodes (you can find more details about master and slave nodes in [15]). All queries are resolved in a distributed manner with a bound of $O(\log_b^2 \log N)$ messages. When a master node receives a query it first checks its own keys to resolve the query, if the lookup is not successful the master node then checks its local finger table. The finger table contains information about 2^{i-1} other master nodes and if the key can be located according to the information stored in the finger table, the query is directly forwarded to the master node storing the data. If the lookup on the local finger table also fails then the master node routes the query to the master node closest to the target according to the finger table. We handle the master node joins/leaves and fails according to join/leave and fail operations respectively presented in [16]. Thus, all the above operations are bounded by $O(\log \log N)$ expected w.h.p. number of messages. Slave nodes do not store information about their neighbors. If a slave node directly receives a query, it checks its own data and if the lookup fails it simply forwards the query to its master node. For simplicity, in the SART proposal we opt for not connecting the slave nodes in a ART arrangement and lookups are not implemented in slave nodes. The master nodes could be thought as "virtual sinks" with an ART overlay between these virtual sinks. Unlike IP in the Internet, the sensor network protocol SP is not at the network layer but instead sits between the network and data-link layer (because data-processing potentially occurs at each hop, not just at end points). Figure 2 shows how P2P overlays can be implemented on top of SP. The P2P overlay (shown as P2P Overlay Management) could be built on top of any generic network protocol. An underlying DHT or Hierarchical Clustering routing protocol (e.g., VRR, CSN, TChord or SNBDT or SART) is recommended as it simplifies the job of overlay management. In particular, it is more efficient to build routing directly on top of the

P2P Architectures	Lookup/update key	Data Overhead-Routing information	Join/Depart Node
Chord	$O(\log N)$	$O(\log N)$ nodes	$O(\log N)$ w.h.p.
H-F-Chord(a)	$O(\log N / \log \log N)$	$O(\log N)$ nodes	$O(\log N)$
LPRS-Chord	$O(\log N)$	$O(\log N)$ nodes	$O(\log N)$
Skip Graphs	$O(\log N)$	$O(1)$	$O(\log N)$ amortized
BATON	$O(\log N)$	Two (2) nodes	$O(\log N)$ w.h.p.
BATON*	$O(\log_m N)$	m nodes	$O(m \log_m N)$
NBDT	$O(\log \log N)$	$O(\log \log N)$ or $2^{2^{i-1}}$ for nodes at level i of left spine	periodical restructuring
ART	$O(\log_b^2 \log N)$	$O(N^{1/4} / \log^c N)$ nodes	$O(\log \log N)$ expected w.h.p.

Table 1: Performance Comparison between ART, NBDT, Chord, BATON and Skip Graphs

link layer instead of implementing it as an overlay on top of a routing protocol [4]. P2P Services and Applications (e.g. event notification, resource allocation, and file systems) can then be built on top of the P2P overlay and sensornet applications could either use these services or communicate with the P2P overlay themselves.

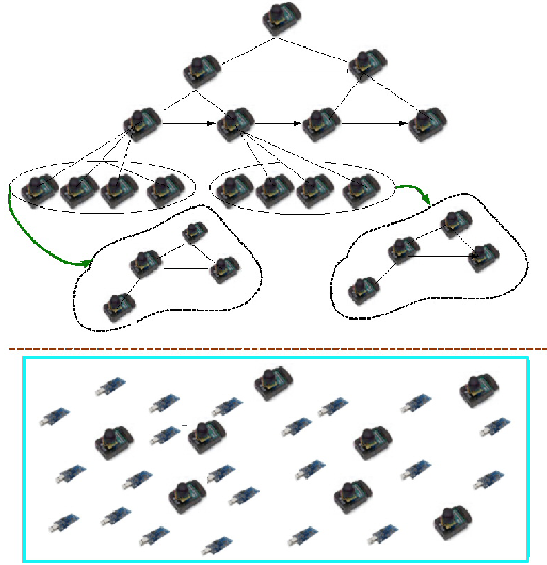


Figure 1: The SART protocol

3. THE SART P2P OVERLAY

Let G a network graph of n sensor nodes and SART the respective overlay of N peers. With each overlay peer p ($1 \leq p \leq N$) we associate a set of pairs $S_p = \{(g, L[g])\}$, where g is a sensor node ($1 \leq g \leq n$) and $L[g]$ its current energy level. The criterion of associating the sensor node g to peer p depends on its current energy level. Obviously, it holds that $N \ll n$. Let's explain more the way we structure our whole system.

One of the basic components of the final SART structure is the LRT (**L**evel **R**ange **T**ree) [16] structure. LRT will be called upon to organize collections of peers at each level of SART.

3.1 The LRT structure [16]: An overview

LRT is built by grouping nodes having the same ancestor

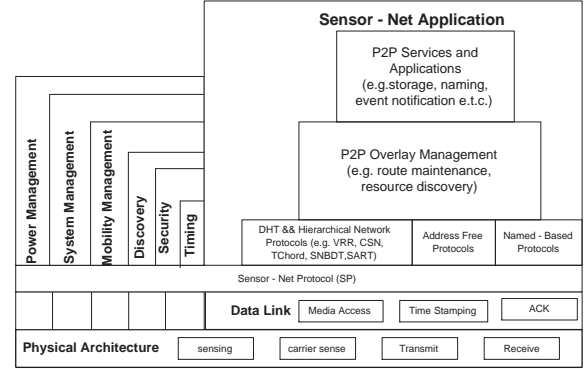


Figure 2: P2P Overlay in SP Architecture

and organizing them in a tree structure recursively. The innermost level of nesting (recursion) will be characterized by having a tree in which no more than b nodes share the same direct ancestor, where b is a double-exponentially power of two (e.g. 2,4,16,...). Thus, multiple independent trees are imposed on the collection of nodes. Figure 3 illustrates a simple example, where $b = 2$.

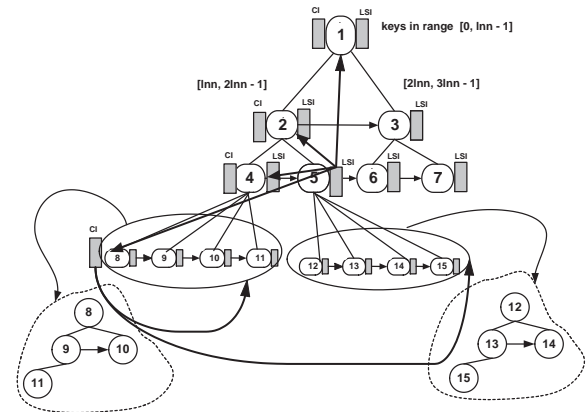


Figure 3: The LRT structure

The degree of the overlay peers at level $i > 0$ is $d(i) = t(i)$, where $t(i)$ indicates the number of peers at level i . It holds that $d(0)=2$ and $t(0)=1$. Let n be w -bit keys. Each peer with label i (where $1 \leq i \leq N$) stores ordered keys that belong in the range $[(i-1) \ln n, i \ln n - 1]$, where $N = n / \ln n$.

is the number of peers. Each peer is also equipped with a table named *Left Spine Index* (LSI), which stores pointers to the peers of the left-most spine (see pointers starting from peer 5). Furthermore, each peer of the left-most spine is equipped with a table named *Collection Index* (CI), which stores pointers to the collections of peers presented at the same level (see pointers directed to collections of last level). Peers having the same father belong to the same collection.

Lookup Algorithm: Assume we are located at peer s and seek a key k . First, the algorithm finds the range where k belongs. If $k \in [(j-1) \ln n, j \ln n - 1]$, it has to search for peer j . The first step of algorithm is to find the LRT level where the desired peer j is located. For this purpose, it exploits a nice arithmetic property of LRT. This property says that for each peer x located at the left-most spine of level i , the following formula holds:

$$\text{label}(x) = \text{label}(\text{father}(x)) + 2^{2^i - 2} \quad (1)$$

For each level i (where $0 \leq i \leq \log \log N$), it computes the value x of its left most peer by applying Equation (1). Then, it compares the value j with the computed value x . If $j \geq x$, it continues by applying Equation (1), otherwise it stops the loop process with current value i . The latter means that node j is located at the i -th level. Then, it follows the i -th pointer of the LSI table located at peer s . Let x the destination peer, that is the leftmost peer of level i . Now, the algorithm must compute the collection in which the peer j belongs to. Since the number of collections at level i equals the number of nodes located at level $(i-1)$, it divides the distance between j and x by the factor $t(i-1)$ and let m the result of this division. Then, it follows the $(m+1)$ -th pointer of the CI table. Since the collection indicated by the $CI[m+1]$ pointer is organized in the same way at the next nesting level, it continues this process recursively.

Analysis: Since $t(i) = t(i-1)d(i-1)$, it gets $d(i) = t(i) = 2^{2^i - 1}$ for $i \geq 1$. Thus, the height and the maximum number of possible nestings is $O(\log \log N)$ and $O(\log_b \log N)$ respectively. Thus, each key is stored in $O(\log_b \log N)$ levels at most and the whole searching process requires $O(\log_b \log N)$ hops. Moreover, the maximum size of the CI and RSI tables is $O(\sqrt{N})$ and $O(\log \log N)$ in worst-case respectively.

Each overlay peer stores tuples $(g, L[g])$, where $L[g]$ is a k -bit key belonging in universe $K = [0, 2^k - 1]$, which represents the current energy-level of the sensor node g . We associate to i^{th} peer the set $S_i = \{(g, L[g])\}$, where $L_g \in [(i-1) \ln K, i \ln K - 1]$. Obviously, the number of peers is $N = K / \ln K$ and the load of each peer becomes $\Theta(\text{polylog} N)$ in expected case with high probability (for more details see [1]). Each energy-level key is stored at most in $O(\log \log N)$ levels. We also equip each peer with the table LSI (Left Spine Index). This table stores pointers to the peers of the left-most spine (for example in figure 3 the peers 1, 2, 4 and 8 are pointed by the LSI table of peer 5) and as a consequence its maximum length is $O(\log \log N)$.

Furthermore, each peer of the left-most spine is equipped with the table CI (Collection Index). CI stores pointers to the collections of peers presented at the same level (see in figure 3 the CI table of peer 8). Peers having same father belong to the same collection. For example in the figure 2, peers 8,9,10 and 11 constitute a collection of peers. It's obvious that the maximum length of CI table is $O(\sqrt{N})$.

3.2 The ART [16] structure: An Overview

The backbone of ART is exactly the same with LRT. During the initialization step the algorithm chooses as cluster_peer representatives the 1st peer, the $(\ln n)$ -th peer, the $(2 \ln n)$ -th peer and so on.

This means that each cluster_peer with label i' (where $1 \leq i' \leq N'$) stores ordered peers with energy-level keys belonging in the range $[(i'-1) \ln^2 n, i' \ln^2 n - 1]$, where $N' = n / \ln^2 n$ is the number of cluster_peers.

ART stores cluster_peers only, each of which is structured as an independent decentralized architecture. Moreover, instead of the Left-most Spine Index (LSI), which reduces the robustness of the whole system, ART introduces the Random Spine Index (RSI) routing table, which stores pointers to randomly chosen (and not specific) cluster_peers (see pointers starting from peer 3). In addition, instead of using fat CI tables, the appropriate collection of cluster_peers can be accessed by using a 2-level LRT structure.

Load Balancing: The join/leave of peers inside a cluster_peer were modeled as the combinatorial game of bins and balls presented in [12]. In this way, for a $\mu(\cdot)$ random sequence of join/leave peer operations, the load of each cluster_peer never exceeds $\Theta(\text{polylog } N')$ size and never becomes zero in expected w.h.p. case.

Routing Overhead: The 2-level LRT is an LRT structure over $\log^{2c} Z$ buckets each of which organizes $\frac{Z}{\log^{2c} Z}$ collections in a LRT manner, where Z is the number of collections at current level and c is a big positive constant. As a consequence, the routing information overhead becomes $O(N^{1/4} / \log^c N)$ in the worst case (even for an extremely large number of peers, let say $N=1.000.000.000$, the routing data overhead becomes 6 for $c=1$).

Lookup Algorithms: Since the maximum number of nesting levels is $O(\log_b \log N)$ and at each nesting level i the standard LRT structure has to be applied in $N^{1/2^i}$ collections, the whole searching process requires $O(\log_b^2 \log N)$ hops. Then, the target peer can be located by searching the respective decentralized structure. Through the polylogarithmic load of each cluster_peer, the total query complexity $O(\log_b^2 \log N)$ follows. Exploiting now the order of keys on each peer, range queries require $O(\log_b^2 \log N + |A|)$ hops, where $|A|$ the answer size.

Join/Leave Operations: A peer u can make a join/leave request at a particular peer v , which is located at cluster_peer W . Since the size of W is bounded by a $\text{polylog} N$ size in expected w.h.p. case, the peer join/leave can be carried out in $O(\log \log N)$ hops.

Node Failures and Network Restructuring: Obviously, node failure and network restructuring operations are according to the decentralized architecture used in each cluster_peer.

3.3 Building the SART Overlay

Let $P_{i,j}$ the j^{th} peer of cluster_peer i . Each overlay peer $P_{i,j}$, stores a set $S_{i,j} = \{(g, L[g])\}$, where $L[g]$ is a k -bit key belonging in universe $K = [0, 2^k - 1]$, which represents the current energy-level of the sensor node g . In particular (and based on design analysis of previous section) it holds that $L[g] \in [(i-1) \ln^2 n, i \ln^2 n - 1]$. Thus, the total set of Cluster_Peer i becomes $S_i = S_{i,1} \cup S_{i,2} \cup \dots \cup S_{i,\Theta(\text{polylog } N)}$, where $|S_{i,j}| \leq n$.

For example in Figure 4, $S_1 = \{(A, L[A]), (C, L[C])\}$

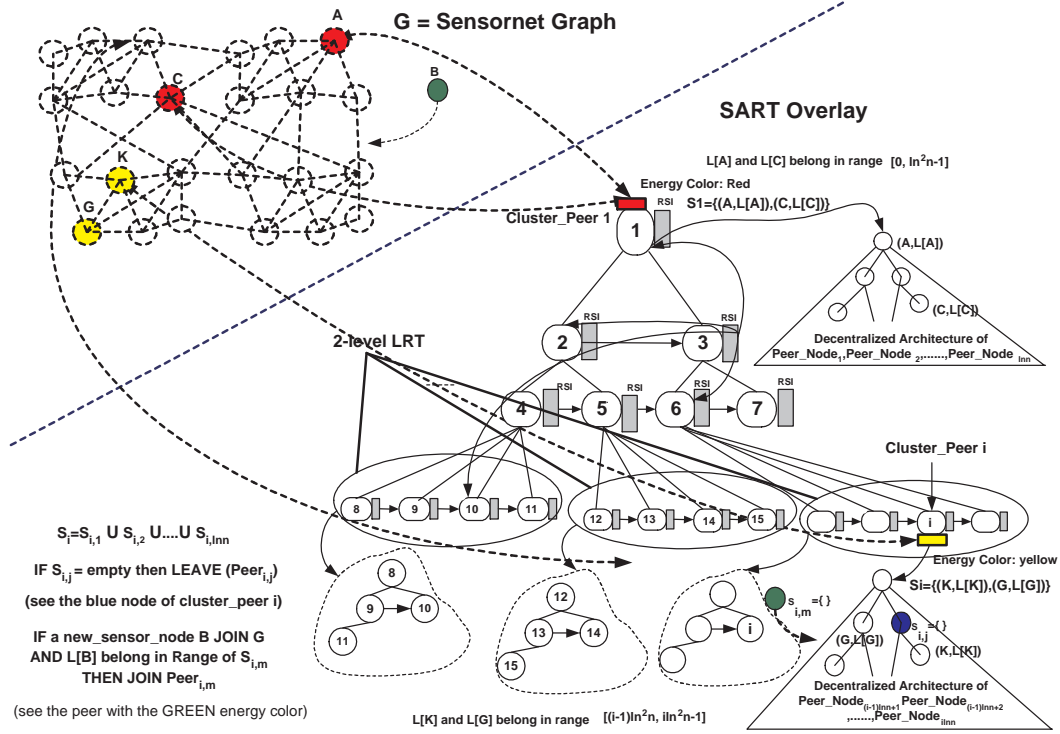


Figure 4: Building the SART Bipartite P2P Overlay

is the set of cluster_peer 1, which stores the energy-level keys of red (energy color) sensors A and C as well as $S_i = \{(K, L[K]), (G, L[G])\}$ is the set of cluster_peer i , which stores the energy-level keys of yellow sensors K and G . Tuples $(A, L[A])$ and $(C, L[C])$ are located in different peers of the decentralized structure associated to cluster_peer 1. The same holds for the tuples $(K, L[K])$ and $(G, L[G])$ in the decentralized structure associated to cluster_peer i .

According to the complexity analysis of ART structure, the theorem 1 follows:

Theorem 1: Assume a SART lookup P2P system for the sensor network G . The queries of the form (a), (b) and (c) require $O(\log_b^2 \log N)$ expected w.h.p. number of messages. The queries of the form (d) and (e) require $O(\log \log N)$ expected w.h.p. number of messages.

Let G the sensor network and T the SART overlay. We are located at sensor node $S \in G$ with low energy level k' and we are looking for a sensor node $R \in G$ with the desired energy level k . Algorithm 1 depicts the pseudocode for the Sensor_Net_Exact_Match_Search routine.

Let G the sensor network and T the SART overlay. We are located at sensor node $S \in G$ with low energy level k' and we are looking for a sensor node $R \in G$ the desired energy level of which belongs in the range $[k_1, k_2]$. Algorithm 2 depicts the pseudocode for the Sensor_Net_Range_Search routine.

Let G the sensor network and T the overlay structure. We are located at sensor node $S \in G$, the energy level of which has been decreased from k_1 to k_2 . We have to find the new overlay peer to which the update node S is going to be associated. Algorithm 3 depicts the pseudocode for the update_overlay_peer routine.

Let G the sensor network and T the overlay structure. If a new sensor node B joins G and $L[B] \in S_{i,m}$ then JOIN $P_{i,m}$ (see the peer with the green energy color). Algorithm 4 depicts the respective pseudocode.

Let G the sensor network and T the overlay structure. If $S_{i,j} = \emptyset$ then LEAVE $P_{i,j}$ (see the blue node of cluster peer i). Algorithm 5 depicts the respective pseudocode.

Algorithm 1 Sensor_Net_Exact_Match_Search(G, S, T, k', k, R)

- 1: Find the peer node to which sensor S (of energy level k') is associated;
 - 2: Let $p \in T$ the respective overlay peer;
 - 3: $r = \text{send_overlay_search}(T, p, k)$; {it is the basic lookup routine of ART structure T }
 - 4: Let $r \in T$ the peer node which stores sensor nodes with the desired energy-level k and let say R a randomly chosen one;
 - 5: Return R
-

Algorithm 2 Sensor_Net_Range_Search(G, S, T, k', k_1, k_2, R)

- 1: Find the peer to which sensor S (of energy level k') is associated;
 - 2: Let $p \in T$ the respective overlay peer;
 - 3: $r = \text{send_overlay_range_search}(T, p, k)$; {it is the range searching routine of ART structure T }
 - 4: Let A the set of peers the desired energy-level of which belong in range $[k_1, k_2]$ and let say R a randomly chosen one;
 - 5: Return R
-

Algorithm 3 Update_Overlay_Peer(G, T, S, k_1, k_2)

- 1: Find the peer to which S is associated according to old energy level k_1 ;
 - 2: Let $p \in T$ the respective overlay peer;
 - 3: Delete (S, k_1) from p ;
 - 4: $r = \text{send_overlay_search}(T, p, k_2)$;
 - 5: Insert the tuple (S, k_2) into r ;
-

Algorithm 4 Join_Overlay_Peer($G, T, B, L[B]$)

- 1: Let $L[B] \in S_{i,m}$ and the m^{th} peer of cluster_peer i does not exist;
 - 2: $\text{send_join_peer}(T, P_{i,m})$; {it is the Join routine of ART structure T }
 - 3: Let $S_{i,m} = \emptyset$ the initial empty set of the new inserted peer $P_{i,m}$;
 - 4: Insert the tuple $(B, L[B])$ into $S_{i,m}$;
-

Algorithm 5 Leave_Overlay_Peer($G, T, P_{i,j}$)

- 1: Let $S_{i,j} = \emptyset$ the empty set of peer $P_{i,j}$;
 - 2: $\text{send_Leave_peer}(T, P_{i,j})$; {it is the Leave routine of ART structure T }
-

4. EXPERIMENTS

For evaluation purposes we used the Distributed Java D-P2P-Sim simulator presented in [16]. The D-P2P-Sim simulator is extremely efficient delivering $> 100,000$ cluster peers in a single computer system, using 32-bit JVM 1.6 and 1.5 GB RAM and full D-P2P-Sim GUI support. When 64-bit JVM 1.6 and 5 RAM is utilized the D-P2P-Sim simulator delivers $> 500,000$ cluster peers and full D-P2P-Sim GUI support in a single computer system.

The Admin tools of D-P2P-Sim GUI (see Figure 5) have specifically been designed to support *reports* on a collection of wide variety of metrics including, protocol operation metrics, network balancing metrics, and even server metrics. Such metrics include frequency, maximum, minimum and average of: number of hops for all basic operations (lookup-insertion-deletion path length), number of messages per node peer (hotpoint-bottleneck detection), routing table length (routing size per node-peer) and additionally detection of network isolation (graph separation). All metrics can be tested using a number of different distributions (e.g. normal, weibull, beta, uniform etc). Additionally, at a system level memory can also be managed in order to execute at low or larger volumes and furthermore execution time can also be logged. The framework is open for the protocol designer to introduce additional metrics if needed. Furthermore, XML rule based *configuration* is supported in order to form a large number of different protocol testing scenarios. It is possible to configure and schedule at once a single or multiple experimental scenarios with different number of protocol networks (number of nodes) at a single PC or multiple PCs and servers distributedly. In particular, when D-P2P-Sim simulator acts in a distributed environment (see Figure 6) with multiple computer systems with network connection delivers multiple times the former population of cluster peers with only 10% overhead.

```
<distribution>
  <random>
    <seed>1</seed>
  </random>
  <beta>
    <alpha>2.0</alpha>
    <beta>4.0</beta>
  </beta>
  <powerLaw>
    <alpha>0.5</alpha>
    <beta>1.0</beta>
  </powerLaw>
</distribution>
```

Figure 7: Snippet from config.xml with the predefined distribution's parameters setup

Our experimental performance studies include a detailed performance comparison with TChord, one of the state-of-the-art P2P overlays for sensor networks. Moreover, we implemented each cluster_peer as a BATON* [10], the best known decentralized tree-architecture. We tested the network with different numbers of peers ranging up to 500,000. A number of data equal to the network size multiplied by 2000, which are numbers from the universe [1..1,000,000,000] are inserted to the network in batches. The synthetic data (numbers) from this universe were produced by the following distributions: beta³, uniform⁴ and power-law⁵. The distribution parameters can be easily defined in configuration file⁶. Also, the predefined values of these parameters are depicted in the figure 7.

For each test, 1,000 exact match queries and 1,000 range queries are executed, and the average costs of operations are taken. Searched ranges are created randomly by getting the whole range of values divided by the total number of peers multiplies α , where $\alpha \in [1..10]$. The source code of the whole evaluation process is publicly available⁷.

In the first tab (see Figure 8) the user can set the number of peers which will constitute the overlay and select the energy level distribution over these nodes. The available distributions are: uniform, normal, beta, and pow-law. After the user has set these two fields then the system's initialization can begin.

In the same tab there is a progress bar so the user can obtain the overall process due to the fact that this process may take several minutes. Also there is a button, which resets the system without the need of closing and reopening the

³<http://acs.lbl.gov/software/colt/api/cern/jet/random/Beta.html>

⁴<http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Random.html>

⁵<http://acs.lbl.gov/software/colt/api/cern/jet/random/Distributions.html#nextPowLaw>

⁶<http://code.google.com/p/d-p2p-sim/downloads/detail?name=Art-config.xml&can=2&q=>

⁷<http://code.google.com/p/d-p2p-sim/>

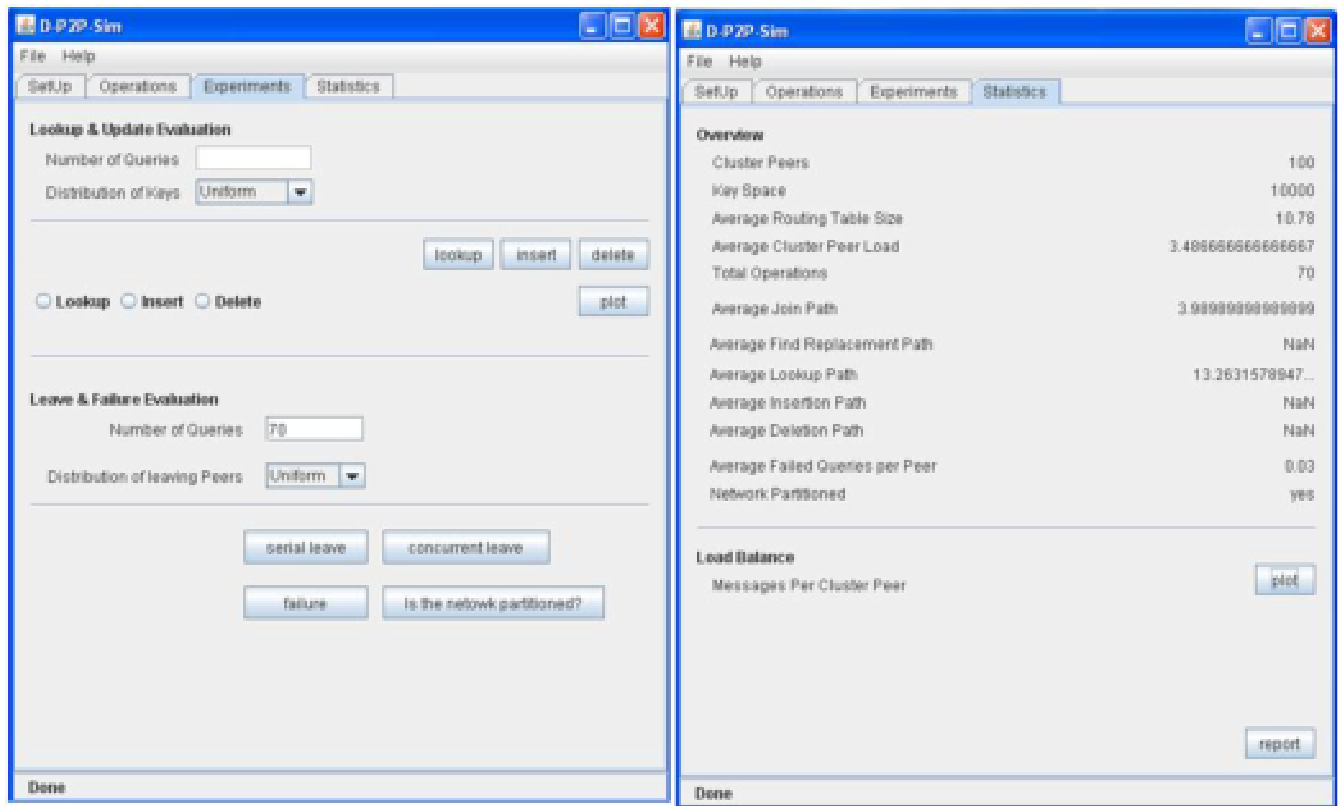


Figure 5: D-P2P-Sim GUI

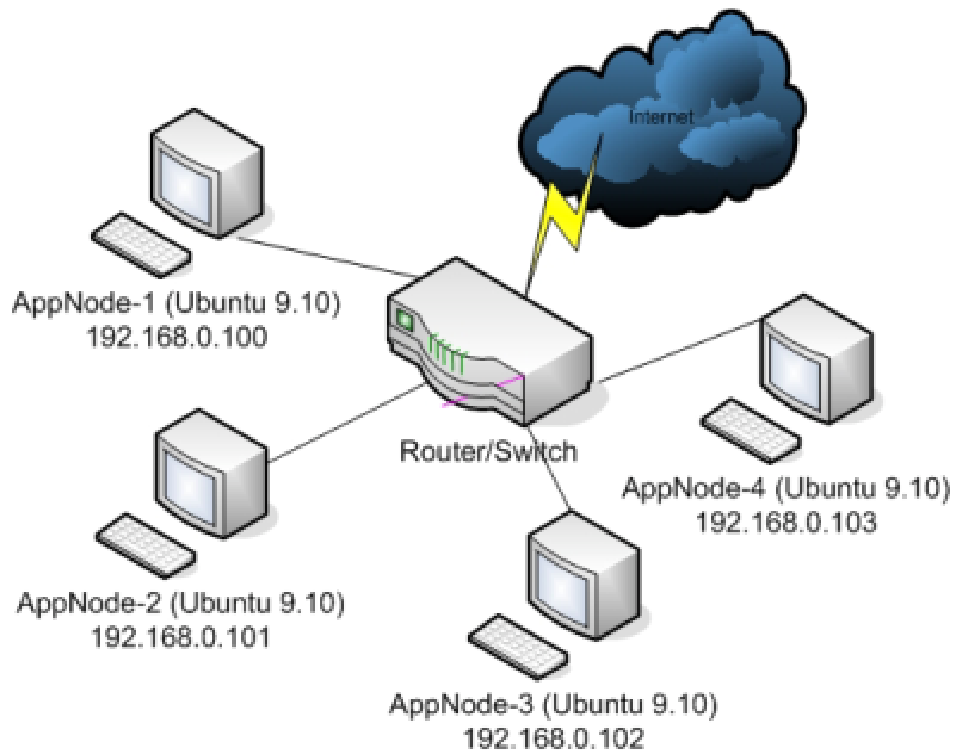


Figure 6: The Distributed Environment

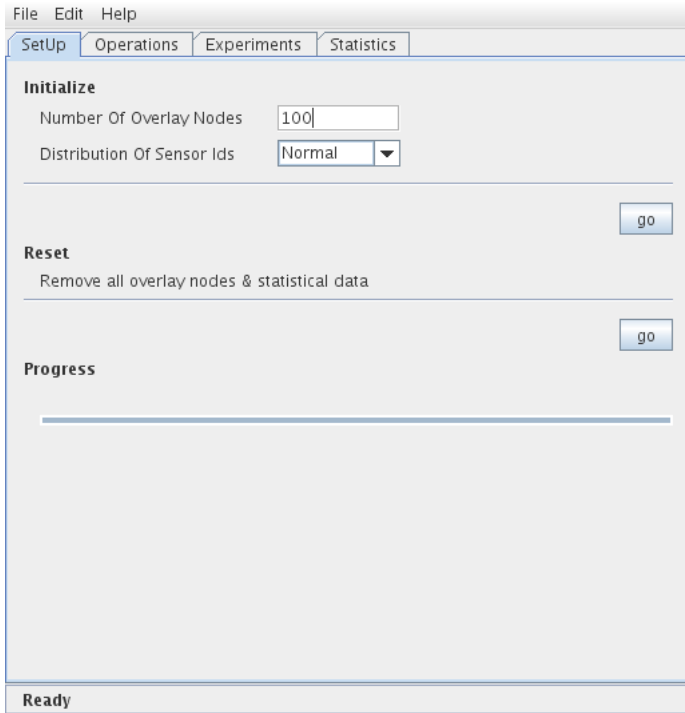


Figure 8: The tab "SetUp"

simulator if we want to carry out several experiments with different number of peers and energy level distribution.

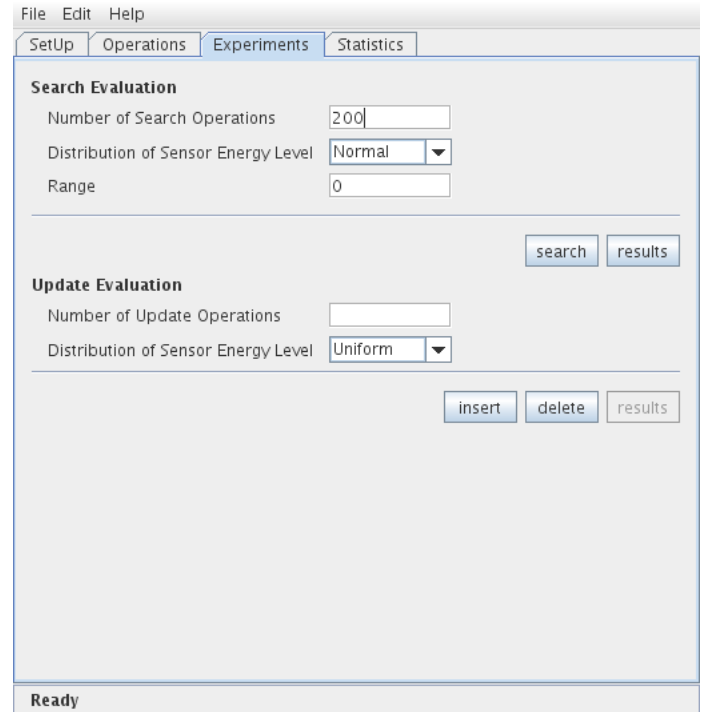


Figure 10: The tab "Experiments"

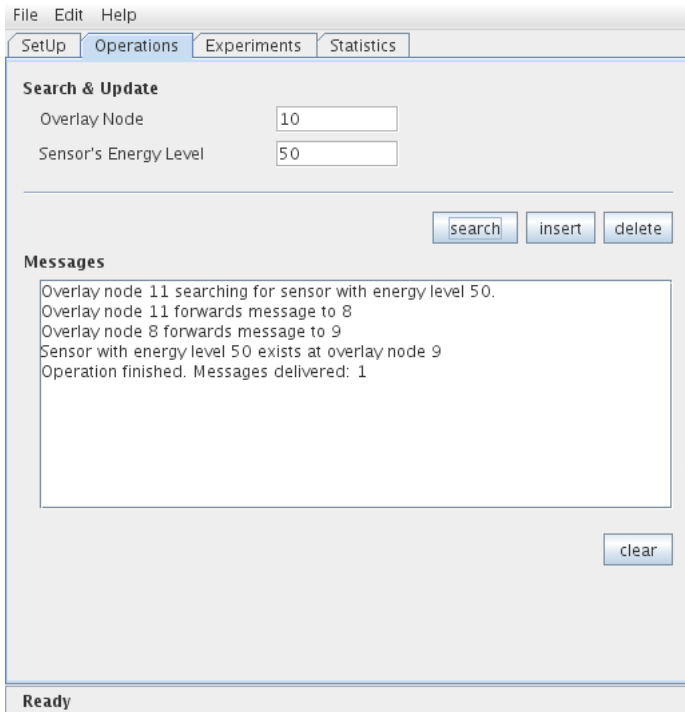


Figure 9: The tab "Operations"

The second tab (see Figure 9) provides the ability to search, insert(join) / delete (leave) and update the energy level of a sensor starting the procedure from any peer in the overlay. While one of these operations is being executed, appropriate messages are appearing at the bottom of this tab.

In the third tab (see Figure 10) the user can prosecute experiments to evaluate the efficiency of the lookup/update operations. There are two panels one for each operation where the user sets the number of the experiments and selects the distribution according to the energy-level keys of the sensors picked up for the experiments. After the termination of the experiments the user can see and save the chart that has been generated. In the forth tab - statistics - the user can see the current number of peers into the system, the number of sensors that have been stored over the peers and the range of sensors' energy level that we can store in the overlay. This tab represents also performance statistics such as the minimum, the maximum and the average path of the total operations that have been performed. Furthermore, this tab generates a chart with the load-balancing over the peers, the number of messages that have been forwarded by each peer)and the number of messages per tree level.

In the most of cases, SART outperforms TChord by a wide margin. As depicted in Figures 11, 12 and 13 our method is almost 2 times faster for $b = 2$, 4 times faster for $b = 4$ and 5 times faster for $b = 16$. As a consequence we have a performance improvement from 50% to 80%.

The results are analogous with respect to the cost of range

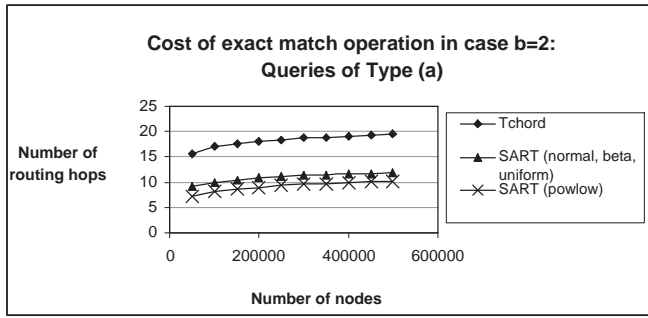


Figure 11: Cost of Exact Match Queries in Case b=2

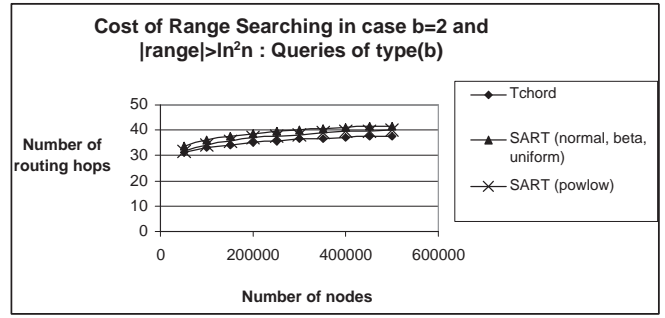


Figure 15: Cost of Range Queries in Case b=2 and $Query_Range_Length > Cluster_Peer_Key_Range$

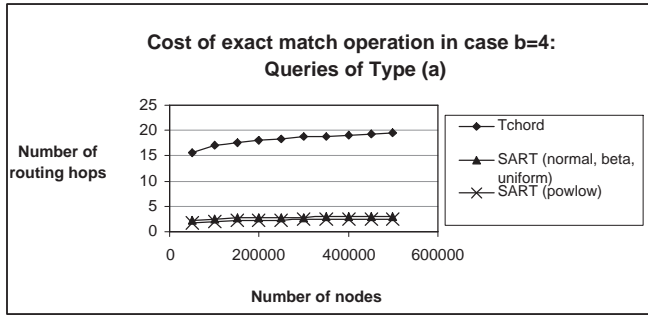


Figure 12: Cost of Exact Match Queries in Case b=4

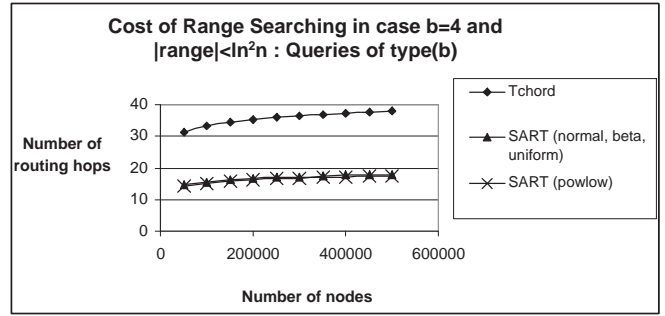


Figure 16: Cost of Range Queries in Case b=4 and $Query_Range_Length < Cluster_Peer_Key_Range$

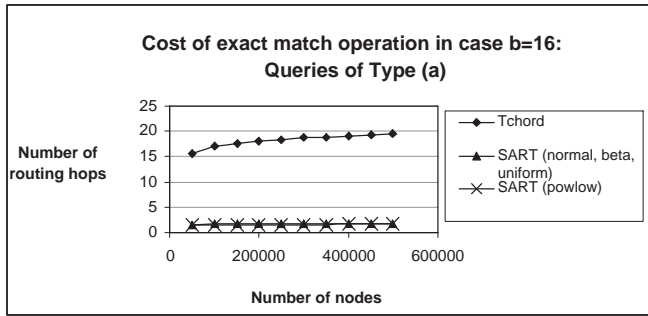


Figure 13: Cost of Exact Match Queries in Case b=16

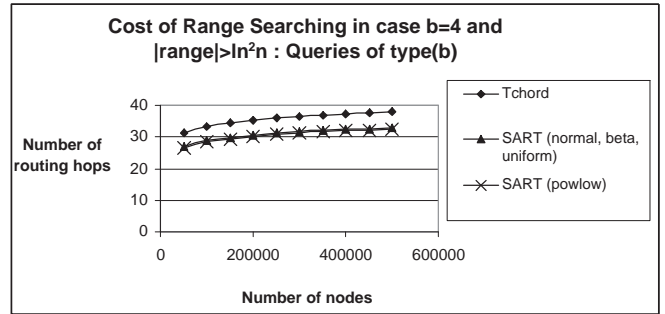


Figure 17: Cost of Range Queries in Case b=4 and $Query_Range_Length > Cluster_Peer_Key_Range$

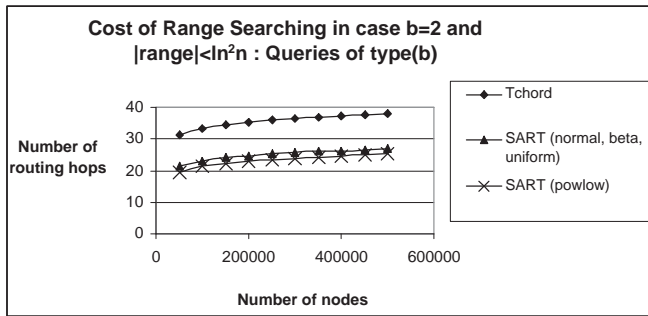


Figure 14: Cost of Range Queries in Case b=2 and $Query_Range_Length < Cluster_Peer_Key_Range$

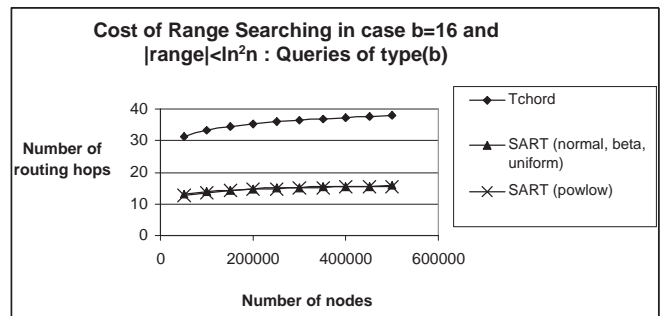


Figure 18: Cost of Range Queries in Case b=16 and $Query_Range_Length < Cluster_Peer_Key_Range$

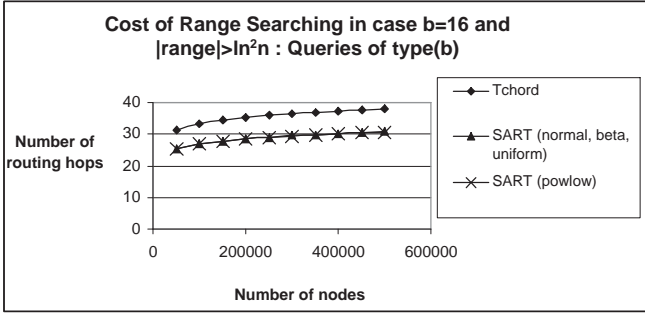


Figure 19: Cost of Range Queries in Case $b=16$ and $Query_Range_Length > Cluster_Peer_Key_Range$

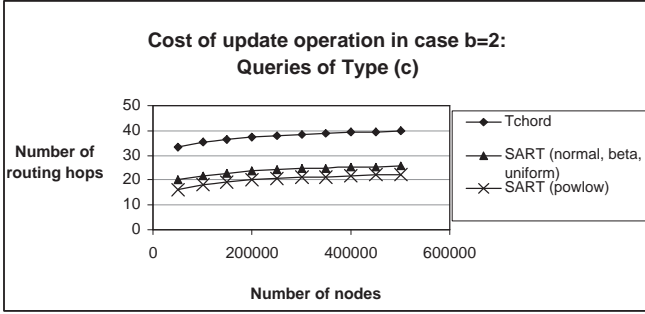


Figure 20: Cost of Update Queries in Case $b=2$

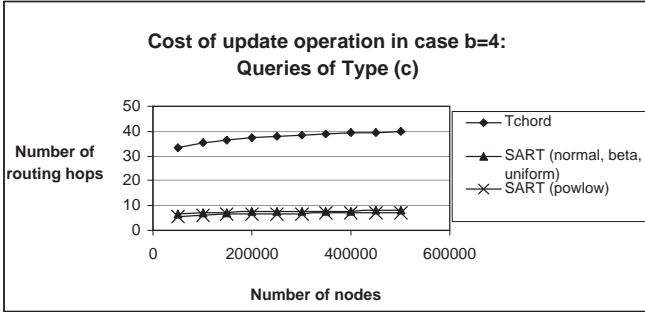


Figure 21: Cost of Update Queries in Case $b=4$

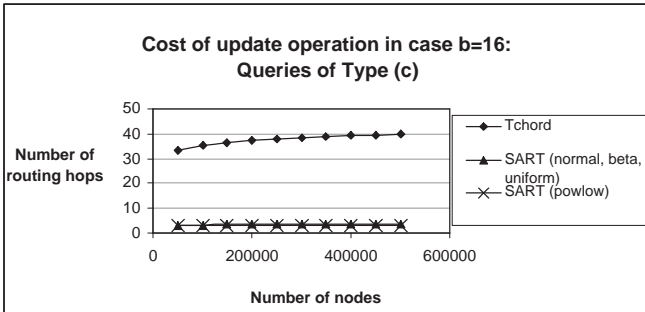


Figure 22: Cost of Update Queries in Case $b=16$

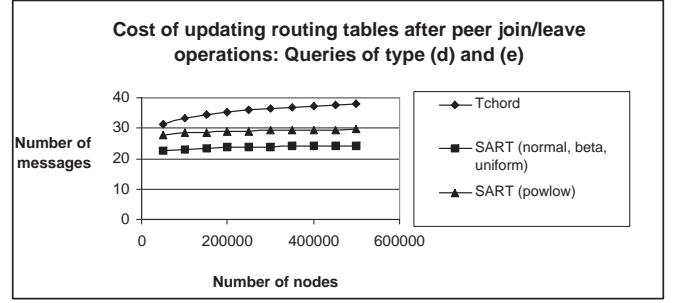


Figure 23: Cost of updating routing tables, after peer join/leave operations: The Cost is independent on parameter b

queries as depicted in Figures 14, 15, 16, 17, 18 and 19.

In case $Query_Range_Length < Cluster_Peer_Key_Range$ and $b = 2$, we have an 25% improvement, however, when $Query_Range_Length > Cluster_Peer_Key_Range$, SART and Tchord have almost similar performance behaviour.

In case $Query_Range_Length < Cluster_Peer_Key_Range$ and $b = 4$, we have an 50% improvement, however, when $Query_Range_Length > Cluster_Peer_Key_Range$ the improvement of our method downgrades to 13.15%.

In case $Query_Range_Length < Cluster_Peer_Key_Range$ and $b = 16$, we have an 52.7% improvement, however, when $Query_Range_Length > Cluster_Peer_Key_Range$ the improvement of our method downgrades to 21.05%.

Figures 20, 21 and 22 depict the cost of update queries. In particular, for $b = 2, 4, 16$, we have an improvement of 37.5%, 75% and 87.5% respectively.

Finally, Figure 23 depicts the cost of updating the routing tables, after peer join/leave operations. For *bad* or non-smooth distributions, like *powlow*, we have an 23.07% improvement. However, for more smooth distributions like *beta*, *normal* or *uniform* the improvement of our method increases to 38.46%.

5. CONCLUSIONS

We considered the problem of constructing efficient P2P overlays for sensornets providing "Energy-Level Application and Services". On this purpose we designed SART, the best-known dynamic P2P overlay providing support for processing queries in a sensornet. We experimentally verified this performance via the D-P2P-Sim framework.

6. REFERENCES

- [1] Muneeb Ali and Koen Langendoen, A Case for Peer-to-Peer Network Overlays in Sensor Networks, International Workshop on Wireless Sensor Network Architecture(WWSNA'07), pages 56-61, Cambridge, Massachusetts, USA, 2007.
- [2] M. Ali and Z. A. Uzmi., CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks. In 2nd CNSR'04, pages 165-174, Fred-ericton, N.B, Canada, 2004.
- [3] J. F. Buford, H. Yu, and E. K. Lua. P2P Networking and Applications. Morgan Kaufman Publications, California, 2008.

- [4] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron., Virtual Ring Routing: Network routing inspired by DHTs. In ACM SIGCOMM'06, pages 351-362, Pisa, Italy, 2006.
- [5] Crainiceanu, A., Linga, P., Gehrke, J. and Shanmugasundaram, J., P-Tree: A P2P Index for Resource Discovery Applications, WWW'04, pages 390-391, New York, NY, USA, 2004.
- [6] D. Clark, C. Partridge, R. T. Braden, B. Davie, S. Floyd, V. Jacobson, D. Katabi, G. Minshall, K. K. Ramakrishnan, T. Roscoe, I. Stoica, J. Wroclawski, and L. Zhang., Making the world (of communications) a different place. ACM SIGCOMM'05 CCR, 35(3):91-96, Philadelphia, PA, 2005.
- [7] M.Demirbas, A.Arora, and M.Gouda., A pursuer-evader game for sensor networks. Sixth Symposium on Self- Stabilizing Systems(SSS'03), pages 1-16, San Francisco, CA, USA, 2003.
- [8] Murat Demirbas, Hakan Ferhatosmanoglu, Peer-to-Peer Spatial Queries in Sensor Networks, IEEE Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pp. 32-40, Linkoping, Sweden, 2003.
- [9] M. Gerla, C. Lindemann, and A. Rowstron., P2P MANET's - new research issues. In Dagstuhl Seminar Proceedings, number 05152, Germany, 2005.
- [10] H. V. Jagadish, B. C. Ooi, K. L. Tan, Q. H. Vu and R. Zhang., Speeding up Search in P2P Networks with a Multi-way Tree Structure, ACM SIGMOD'06, pages 1-12, Chicago, Illinois, 2006.
- [11] H. V. Jagadish, B. C. Ooi, and Q. H. Vu., Baton: A balanced tree structure for peer-to-peer networks. In Proceedings of the 31st VLDB'05 Conference, pages 661-672, Trondheim, Norway, 2005.
- [12] A. Kaporis, C. Makris, S. Sioutas, A. Tsakalidis, K. Tschlas, and C. Zaroliagis. Improved Bounds for Finger Search on a RAM. *Algorithms*, Vol. 2832:325-336, 2003.
- [13] A.Mainwaring, J.Polastre, R.Szewczyk, D.Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. ACM Int. Workshop on Wireless Sensor Networks and Applications, September 2002.
- [14] S.Sioutas, NBDT: An efficient p2p indexing scheme for web service discovery, Journal of Web Engineering and Technologies, Vol. 4 (1), pp 95-113, 2008.
- [15] S. Sioutas, K. Oikonomou, G. Papaloukopoulos, M. Xenos, Y. Manolopoulos, "An Optimal Bipartite P2P Overlay for Energy-Level Queries in Sensor Networks", Proceedings of the ACM international Conference on Management of Emergent Digital Ecosystems - ACM Special Interest Group on Applied Computing (ACM-SIGAPP MEDES 2009), Lyon, France, pp.361-368.
- [16] S.Sioutas, G. Papaloukopoulos, E. Sakkopoulos, K. Tschlas, Y. Manolopoulos and P. Triantafillou "Brief Announcement: ART:Sub-Logarithmic Decentralized Range Query Processing with Probabilistic Guarantees", In Proceedings of Twenty-Ninth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (ACM PODC 2010), Zurich, Switzerland July 25-28, pp. 118-120, 2010.